Homework 2

Kevin Leary

**Problem 1**

Psuedo-code:

Ask user for limit for Pythagorean triple

Create variables to be used throughout program

While C^2 is less than the limit

For each m in the range of 1 to n+1

Do calculations for Pythagorean triple

If a b or c is 0 then

End loop

If a b or c is divisible by a primitive pythagoren triple

End loop

Print a b and c

Increment n

Set variables x and c

While c is less than the limit entered

Calculate all the primitives

Print a b and c

Increment x

limit=int(input("Enter upper limit: "))

c=**0**

n=**2**

**while**(c<limit): #as long as C is lower then the limit

**for** m **in** range(**1**,n+**1**):

a=n\*n - m\*m

b=**2**\*n\*m

c=n\*n+m\*m

**if**(c>limit): #goes til limit

**break**

**if**(a==**0** **or** b==**0** **or** c==**0**): #wont display a 0

**break**

**if**((a%**3** == **0** **and** b%**4**==**0** **and** c%**5** == **0**) **or** (a%**4** == **0** **and** b%**3** == **0** **and** c%**5** == **0**)): #wont duplicate a primitive triple

**break**

**print**(a,b,c)

n=n+**1**

x = **1**

c = **0**

**while**(c<limit): #for primitive triples

a = x\***3**

b = x\***4**

c = x\***5**

**print**(a,b,c)

x = x+**1**

**Problem 2**

In function find\_dup\_str

For each character in the string s

Set variable og\_string to s with splicing

If the length of the substring is less than the given length

Return “”

For each character in the string s

Set variable dup\_string to s with correct splicing

If og\_string and dup\_string are equal

return one of them

If the length is not the same between og and dup

Then stop comparing them

If og and dup are equal

Return one of them

Out of function find\_dup\_str

Ask user for a string and a length

Print find\_dup\_str

In function find\_max\_dup

Set iter to one less than the length

While iter is greater than 0

If function find\_dup\_str does not equal “”

then end loop

decrement iter

return find\_dup\_str

call functions

**def** find\_dup\_str(s, n):

*#find the first letter in the string then compare to the others*

og\_string = ''

dup\_string = ''

**for** i **in** list(range(len(s))): *#iterate through string*

og\_string = s[i:i+n:1]

*#j = n+i #alwasy starts right after og\_string*

**if**(len(s[i:i+n:1]) < n): *#case for if the string is less then n*

**return** ""

**for** j **in** list(range(n+i, len(s))):

dup\_string = s[j:j+n:1]

*#print(s[j:j+n:1])*

**if**(og\_string == dup\_string):

**return** og\_string

**if**(len(og\_string) != len(dup\_string)): *# when it stops being same len stop comparing*

**return** ""

**if**(og\_string == dup\_string):

**return** og\_string

**if**(og\_string != dup\_string **and** i == (len(s) - 1)):

**return** ""

**def** find\_max\_dup(s):

i = int(len(s) - 1) *#i starts at second element as it becomes n*

**while**(i > 0): *#increment through*

**if**(find\_dup\_str(s, i) != ""):

**break**

i = i-1

**return** find\_dup\_str(s, i)

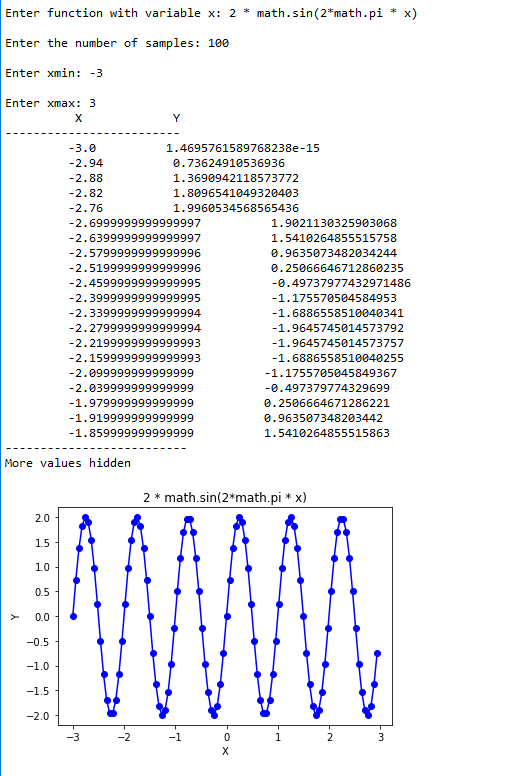
s = input("Enter a string to find dups: ")

*#n = int(input("Enter the length of duplicate: "))*

*#print(find\_dup\_str(s,n))*

**print**(find\_max\_dup(s))

**Problem 3**



**import** pylab

fun\_str = input("Enter function with variable x: ")

ns = int(input("Enter the number of samples: "))

xmin = float(input("Enter xmin: "))

xmax = float(input("Enter xmax: "))

xrange = (xmax - xmin) / ns

x = xmin

i = 0

xs = list()

ys = list()

**while**(x <= xmax):

xs.append(x)

x += xrange

**for** x **in** xs:

y = eval(fun\_str)

ys.append(y)

**print**(" X Y")

**print**("-------------------------")

**while**(i < 20):

**print**(" {0} {1}".format(xs[i], ys[i]))

i= i+1;

**print**("--------------------------")

**print**("More values hidden")

pylab.plot(xs,ys, "bo-")

pylab.xlabel("X")

pylab.ylabel("Y")

pylab.title(fun\_str)

**Problem 4**

def input\_tuple(prompt, types, sep):

try:

new\_tuple = tuple()

info = input(prompt) #setting up user given info

info\_extra = info.split(sep)

if len(info\_extra) != len(types):

return new\_tuple #returns empty tuple

else:

for i in range(len(types)):

new = types[i](info\_extra[i])

new\_tuple.append(new)

new\_tuple = tuple(new\_tuple)

return new\_tuple

except ValueError:

print("Wrong parameters entered")

return ()

#part B

def input\_tuple\_lc(prompt, types, sep):

try:

new\_list=[]

info = input(prompt)

info\_extra = info.split(sep)

if len(info\_extra) != len(types):

return new\_list #returns empty list

else:

new\_list=[types[i](info\_extra[i]) for i in range(len(types))]

new\_tuple=tuple(new\_list)

return new\_tuple

except ValueError:

print("Wrong parameters entered")

return ()

#part C

def read\_tuple(file\_obj, types, sep): #extra

try:

new\_list=[]

new\_list=[types[i](file\_obj[i]) for i in range(len(types))] #reading from the file

new\_tuple=tuple(new\_list)

return new\_tuple

except ValueError:

print("Wrong parameters entered")

return ()

call1 = input\_tuple("Enter first name, last name, age (float), ID (int), fulltime (bool): ", (str, str, float, int, bool), ',')

call2 = input\_tuple("Enter first name, last name, age (float), ID (int), fulltime (bool): ", (str, str, float, int, bool), ',')

f = open(“cars.csv”, “r”)

for line in f:

cars=read\_tuple(line, (str, str, float, int, bool), ',')

print(cars)

**Problem 5**

#Part A

numbers = [4, -3, 0, 2, -1, 5]

numbers\_squared = ['y\*y=' + str(n\*n) for n in numbers]

#print(numbers\_squared)

#part B

numbers\_solution = ['solution #' + str(n+1) + '=' + str(numbers[n]\*\*2) for n in range(len(numbers))]

print(numbers\_solution)

#part C

lst = ["zero", "one", "two", "three"]

new\_lst = [str(i) + ' ' + lst[i] for i in range(len(lst))]

print(new\_lst)

#part D

a = ['a','b','c']

b = [1,2]

cartesian = []

for x, y in [(x,y) for x in a for y in b]:

cartesian += (x,y)

print(cartesian)

#part E

lst1 = [56, 25, 8, 11, 16, 20, 18, 50, 7, 42]

lst2 = [5, 3, 6]

**Problem 6**

def get\_csv\_data(f, string\_pos\_lst, sep=","):

lebron\_lst = []

f.readline()

for line in f: #iterate through

line\_list = line.split(sep)

if(string\_pos\_lst in line\_list): #special str values entered

lebron\_tuple = (str(string\_pos\_lst))

lebron\_list.append(lebron\_tuple)

return lebron\_lst

#part B

def get\_columns(lebron\_lst, cols\_lst):

col\_data = []

for column in cols\_lst:

try:

index = lebron\_lst[0].index(column)

tmp\_col\_data = []

for lbj in lebron\_lst[1:]: #iterate through list with a splice

tmp\_col\_data.append(lbj[index])

col\_data.append(tmp\_col\_data)

except ValueError:

print("Wrong columns taken")

pass #keep it going

return col\_data

bb\_file = open(“lb-james.csv”, “r”) #error here for some reason

james\_lst = get\_csv\_data(bb\_file, [0, 2, 3, 4], “,”)

print(james\_lst)

selected\_cols\_lst = get\_columns(james\_lst, ["Season","Age","PTS"])

selected\_col\_list = get\_columns(james\_list, ["Season", "3P%", "2P%", "FT%"])

x\_axis = [int(x.split("-")[0]) for x in selected\_col\_list[0]]

y1 = selected\_col\_list[1]

y2 = selected\_col\_list[2]

y3 = selected\_col\_list[3]

pylab.plot(x\_axis, y1, '-b', label="3-point precentage")

pylab.plot(x\_axis, y2, '-r', label="2-point precentage")

pylab.plot(x\_axis, y3, '-g', label="free throw precentage")

pylab.title("Lebron vs Time")

pylab.xticks(x\_axis, x\_axis)

pylab.xlabel("Year")

pylab.ylabel("Percentage")

pylab.ylim(0, 1.0)